IN THE CLAIMS:

1. (Currently amended) A method of locking a system resource in a multiprocessor system, comprising:

attempting to obtain a lock on the system resource;

associating a hand-off lock with the lock on the system resource if the attempt to obtain the lock is unsuccessful, wherein the hand-off lock includes a plurality of per-processor spin ~~field for each of the multiprocessor system~~ fields, each per-processor spin field being associated with one processor in the multiprocessor system and identifying a memory location dedicated to that one processor, and wherein only the one processor may spin on the memory location identified by the processor's corresponding dedicated per-processor spin field at any one time; and

obtaining the hand-off lock on the system resource if the attempt to obtain the lock on the system resource is unsuccessful, wherein obtaining the hand-off lock includes spinning on [[a]] the memory location identified by the per-processor spin field for an associated processor.

2. (Original) The method of claim 1, wherein the lock is a simple lock.

3. (Original) The method of claim 1, wherein the hand-off lock is a krlock.

4. (Original) The method of claim 1, wherein the step of attempting to obtain a lock on the system resource is performed a predetermined number of times before associating a hand-off lock with the lock on the system resource.

5. (Original) The method of claim 1, wherein the hand-off lock is obtained from a pool of hand-off locks.

6. (Original) The method of claim 1, wherein associating a hand-off lock with the lock on the system resource includes storing an index of the hand-off lock in a lock word of the lock on the system resource.

7. (Original) The method of claim 1, wherein if the lock on the system resource is freed, the method further comprises:

obtaining the lock on the system resource;

releasing the hand-off lock; and

handing-off the hand-off lock to a next processor spinning on the hand-off lock.

8. (Previously presented) The method of claim 1, wherein the method is implemented in a multiprocessor system having one or more nodes, and wherein the hand-off lock includes a per-node word which contains a state of the hand-off lock on each node of the multiprocessor system.

9. (Original) The method of claim 8, wherein when the lock on the system resource is released, the per-node word and per-processor spin fields of the hand-off lock are updated to reflect a next processor obtaining the lock on the system resource.

10. (Original) The method of claim 1, wherein the method is implemented in one of a SMP, a NUMA, and a ccNUMA system.

11. (Currently amended) A computer program product in a computer readable medium for locking a system resource in a multiprocessor system, comprising:

first instructions for attempting to obtain a lock on the system resource;

second instructions for associating a hand-off lock with the lock on the system resource if the attempt to obtain the lock is unsuccessful, wherein the hand-off lock includes a plurality of per-processor spin ~~field for each of the multiprocessor system~~ fields, each per-processor spin field being associated with one processor in the multiprocessor system and identifying a memory location dedicated to that one processor, and wherein only the one processor may spin on the memory location identified by the processor's corresponding dedicated per-processor spin field at any one time; and

third instructions for obtaining the hand-off lock on the system resource if the attempt to obtain the lock on the system resource is unsuccessful, wherein obtaining the

hand-off lock includes spinning on [[a]] the memory location identified by the per-processor spin field for an associated processor.

12. (Original) The computer program product of claim 11, wherein the lock is a simple lock.

13. (Original) The computer program product of claim 11, wherein the hand-off lock is a krlock.

14. (Previously presented) The computer program product of claim 11, further comprising instructions for executing the first instructions a predetermined number of times before executing the second instructions.

15. (Original) The computer program product of claim 11, wherein the hand-off lock is obtained from a pool of hand-off locks.

16. (Original) The computer program product of claim 11, wherein the second instructions include instructions for storing an index of the hand-off lock in a lock word of the lock on the system resource.

17. (Original) The computer program product of claim 11, wherein the computer program product further comprises:

fourth instructions for obtaining the lock on the system resource, if the lock on the system resource is freed;

fifth instructions for releasing the hand-off lock; and

sixth instructions for handing-off the hand-off lock to a next processor spinning on the hand-off lock.

18. (Previously presented) The computer program product claim 11, wherein the hand-off lock includes a per-node word which contains a state of the hand-off lock on each node of a multiprocessor system.

19. (Original) The computer program product of claim 18, further comprising fourth instructions for updating the per-node word and per-processor spin fields of the hand-off lock to reflect a next processor obtaining the lock on the system resource, when the lock on the system resource is released.

20. (Original) The computer program product of claim 11, wherein the first, second and third instructions are formatted for use with one of an SMP, a NUMA, and a ccNUMA system.

21. (Currently amended) An apparatus for locking a system resource in a multiprocessor system, comprising:

means for attempting to obtain a lock on the system resource;

means for associating a hand-off lock with the lock on the system resource if the attempt to obtain the lock is unsuccessful, wherein the hand-off lock includes a plurality of per-processor spin ~~field for each of the multiprocessor system~~ fields, each per-processor spin field being associated with one processor in the multiprocessor system and identifying a memory location dedicated to that one processor, and wherein only the one processor may spin on the memory location identified by the processor's corresponding dedicated per-processor spin field at any one time; and

means for obtaining the hand-off lock on the system resource if the attempt to obtain the lock on the system resource is unsuccessful, wherein obtaining the hand-off lock includes spinning on [[a]] the memory location identified by the per-processor spin field for an associated processor.

22. (Original) The apparatus of claim 21, wherein the lock is a simple lock.

23. (Original) The apparatus of claim 21, wherein the hand-off lock is a krlock.

24. (Original) The apparatus of claim 21, wherein the means for attempting to obtain the lock on the system resource operates a predetermined number of times before the means for associating the hand-off lock operates.

25. (Original) The apparatus of claim 21, wherein the hand-off lock is obtained from a pool of hand-off locks.

26. (Original) The apparatus of claim 21, wherein the means for associating the hand-off lock includes means for storing an index of the hand-off lock in a lock word of the lock on the system resource.

27. (Original) The apparatus of claim 21, wherein the apparatus further comprises:

means for obtaining the lock on the system resource, if the lock on the system resource is freed;

means for releasing the hand-off lock; and

means for handing-off the hand-off lock to a next processor spinning on the hand-off lock.

28. (Previously presented) The apparatus claim 21, wherein the apparatus is part of a multiprocessor system having one or more nodes, and wherein the hand-off lock includes a per-node word which contains a state of the hand-off lock on each node of the multiprocessor system.

29. (Original) The apparatus of claim 28, further comprising means for updating the per-node word and per-processor spin fields of the hand-off lock to reflect a next processor obtaining the lock on the system resource, when the lock on the system resource is released.

30. (Original) The apparatus of claim 21, wherein the apparatus is part of one of a SMP, a NUMA, and a ccNUMA system.

31. (New) The method of claim 1, wherein the only processor in the multiprocessor system that may obtain the lock is a processor that currently owns the hand-off lock.

32. (New) The method of claim 1, wherein a state structure associated with the hand-off lock identifies an order in which processors spinning on memory locations identified by the per-processor spin fields of the hand-off lock are to be given ownership of the hand-off lock.

33. (New) The method of claim 32, wherein when a processor that owns the hand-off lock acquires the lock, the processor that owns the hand-off lock hands ownership of the hand-off lock to a next processor attempting to acquire the hand-off lock as determined by the state structure.

34. (New) The method of claim 32, wherein the state structure identifies which processor of the multiprocessor system owns the hand-off lock and which processors of the multiprocessor system are trying to obtain the hand-off lock, if any.

35. (New) The method of claim 1, wherein data in the memory locations identified by the per-processor spin fields is maintained in corresponding caches of the associated processors while the processors spin on their corresponding memory locations.

36. (New) The method of claim 1, wherein the hand-off lock serializes handing off of ownership of the hand-off lock to processors in the multiprocessor system based on the per-processor spin fields.

37. (New) The computer program product of claim 11, wherein the only processor in the multiprocessor system that may obtain the lock is a processor that currently owns the hand-off lock.

38. (New) The computer program product of claim 11, wherein the hand-off lock serializes handing off of ownership of the hand-off lock to processors in the multiprocessor system based on the per-processor spin fields.

39. (New) The computer program product of claim 11, wherein a state structure associated with the hand-off lock identifies an order in which processors spinning on memory locations identified by the per-processor spin fields of the hand-off lock are to be given ownership of the hand-off lock.

40. (New) The computer program product of claim 39, wherein when a processor that owns the hand-off lock acquires the lock, the processor that owns the hand-off lock hands ownership of the hand-off lock to a next processor attempting to acquire the hand-off lock as determined by the state structure.

41. (New) The computer program product of claim 39, wherein the state structure identifies which processor of the multiprocessor system owns the hand-off lock and which processors of the multiprocessor system are trying to obtain the hand-off lock, if any.

42. (New) The apparatus of claim 21, wherein data in the memory locations identified by the per-processor spin fields is maintained in corresponding caches of the associated processors while the processors spin on their corresponding memory locations.

43. (New) The apparatus of claim 21, wherein the only processor in the multiprocessor system that may obtain the lock is a processor that currently owns the hand-off lock.

44. (New) The apparatus of claim 21, wherein the hand-off lock serializes handing off of ownership of the hand-off lock to processors in the multiprocessor system based on the per-processor spin fields.

45. (New) The apparatus of claim 21, wherein a state structure associated with the hand-off lock identifies an order in which processors spinning on memory locations identified by the per-processor spin fields of the hand-off lock are to be given ownership of the hand-off lock.

46. (New) The apparatus of claim 45, wherein when a processor that owns the hand-off lock acquires the lock, the processor that owns the hand-off lock hands ownership of the hand-off lock to a next processor attempting to acquire the hand-off lock as determined by the state structure.

47. (New) The apparatus of claim 45, wherein the state structure identifies which processor of the multiprocessor system owns the hand-off lock and which processors of the multiprocessor system are trying to obtain the hand-off lock, if any.

48. (New) The apparatus of claim 21, wherein data in the memory locations identified by the per-processor spin fields is maintained in corresponding caches of the associated processors while the processors spin on their corresponding memory locations.